Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

An Incremental Multi-Centroid, Multi-Run Sampling Scheme for *k*-medoids-based algorithms

S-C. Chu[1], J. F. Roddick[1] and J-S. Pan[2]
[1]*School of Informatics and Engineering, Flinders University of South Australia, Adelaide, Australia.*
[2]*Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan.*

Abstract

Data clustering has become an important task for discovering significant patterns and characteristics in large spatial databases. The *Multi-Centroid, Multi-Run Sampling Scheme* ($MCMRS$) has been shown to be effective in improving the k-medoids-based clustering algorithms in our previous work. In this paper, a more advanced sampling scheme termed *Incremental Multi-Centroid, Multi-Run Sampling Scheme* ($IMCMRS$) is proposed for k-medoids-based clustering algorithms. Experimental results demonstrate the proposed scheme can not only reduce by more than 80% computation time but also reduce the average distance per object compared with $CLARA$ and $CLARANS$. $IMCMRS$ is also superior to $MCMRS$.

1 Introduction

Clustering is a useful practice of classification imposed over a finite set of objects. The goal of clustering is to group sets of objects into classes such that single groups have similar characteristics, while dissimilar objects are in separate groups.

Various existing clustering algorithms have been proposed and designed to fit various formats and constraints of application including k-means [16], k-medoids [11], $BIRCH$ [18], $CURE$ [8], $CHAMELEON$ [10], $DBSCAN$ [4],

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

AUTOCLUST [6] and *AUTOCLUST*$^+$ [5]. No single algorithms is suitable for all types of objects, nor are all algorithms appropriate for all problems, however, k-medoids algorithms have been shown to be robust to outliers (or noise) and are not generally influenced by the order of presentation of objects. Moreover, k-medoids algorithms are invariant to translations and orthogonal transformations of objects. Partitioning Around Medoids (*PAM*) [11], Clustering LARge Applications (*CLARA*) [11], Clustering LARge Applications based on RANdomized Search (*CLARANS*)[17], Clustering Large Applications based on Simulated Annealing (*CLASA*), fuzzy k-medoids algorithm [13], genetic k-medoids algorithm [15] are existing k-medoids-based algorithms.

Both k-means (which adopts as the representative point the weighted mean of the cluster) and k-medoids (which adopts as the representative point the most central object in the cluster) algorithms are a partitioning method. One of the main elements to limit the use of k-medoids algorithm is the inefficiency of k-medoids algorithms comparing with k-means - k-means algorithm is several orders of magnitude faster than k-medoids algorithm. In general, it is not efficient to use k-medoids algorithm even for moderate sized datasets. This shortcoming can be overcome with the aid of an efficient sampling scheme.

In the following section we discuss existing k-medoids-based algorithms including the Multi-Centroid, Multi-Run Sampling Scheme (*MCMRS*) discussed in [2]. We then propose improvements to *MCMRS* [2] by adopting the adaptive concept - Incremental Multi-Centroid, Multi-Run Sampling scheme (*IMCMRS*) which will be described in Section 3 while Section 4 will report on some promising results we have obtained by using four artificial databases and one image database. The conclusions are given in Section 5.

2 Related Work

2.1 *PAM*

PAM (Partitioning Around Medoids) was developed to find the k most representative objects (medoids) that represent k clusters such that non-selected objects are clustered with the medoid to which it is the most similar. The total distance between non-medoid objects and their representative medoid may be reduced by swapping one of the medoids with one of the objects iteratively. Obviously, it is time consuming even for the moderate number of objects and small number of medoids.

2.2 *CLARA*

CLARA (Clustering LARge Applications) was developed to overcome the computational complexity drawbacks of *PAM*. Instead of finding representative objects for the whole data set, *CLARA* reduces the complexity by drawing multiple samples of the objects and applying *PAM* on each sample. The final medoids are obtained from the best result of these multiple draws.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

2.3 *CLARANS*

The clustering process in *CLARANS* (Clustering Large Applications based on RANge Search) is formalised as searching through a graph where each node is represented by a set of k medoids – two nodes are neighbours if they only differ by one medoid. *CLARANS* starts with a randomly selected node. It moves to the neighbour node if a test for the *maxneighbour* number of neighbours is successful; otherwise it records the current node as a local minimum. If the node is found to be a local minimum, it restarts with a new randomly selected node and repeats the search for a new local minimum. *CLARANS* also displays the drawback in that uses a randomised search algorithm, the efficiency is still slow although it is faster than *CLARA* and *PAM*.

2.4 *CLASA*

CLASA (Clustering Large Applications based on Simulated Annealing) generates medoids by applying the simulated annealing method [12, 9]. The collection of the k medoids is called a *state* in *CLASA* [1]. There are $\frac{T!}{k!(T-k)!}$ states where T is the total number of objects. It is possible to move from current state to any other states depending on the moving strategy. For our preliminary experiments, we consider only movements between two states that involve changing only one medoid. The *CLASA* algorithm can be illustrated as follows:

1. Choose an initial state s of medoids at random and set the initial temperature $Temp = T_0$.
2. Randomly choose another state s' (some perturbation of state s) by swapping the medoids with the objects. Calculate the difference of total distortion $\Delta D = D_t(s') - D_t(s)$. If $\Delta D < 0$, replace the state s by s' otherwise replace s by s' with probability $e^{\frac{-\Delta D}{Temp}}$ and go to step 3.
3. If the number of total distance drops *dis_drop* exceeds a prescribed number or the fixed number of perturbations *per* is reached, go to step 4; otherwise go to step 2.
4. Terminate the program and return the selected medoids if the temperature $Temp$ is below some prescribed freezing temperature T_f or the total number of perturbations *total_per* is reached; otherwise lower the temperature $Temp$ and go to step 2.

There are several possible methods for the annealing schedule, it is convenient to set $Temp = T_0\eta^t$, where t is the number of iterations, η is a constant coefficient, $0 < \eta < 1$.

2.5 Fuzzy k-medoids algorithms

The concept of fuzzy k-means clustering algorithm can be applied to generate the medoids. Two methods, the fuzzy k-medoids algorithm and fuzzy k-trimmed medoids algorithm were proposed in [13]. The evaluation function of

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

the fuzzy k-medoids algorithm is to minimise $J_m(O,X) = \sum_{j=1}^{T}\sum_{i=1}^{k} u_{ij}^m r(x_j, o_i)$ where $r(x_j, o_i)$ is the dissimilarity between object x_j and medoid o_i.

2.6 Genetic k-medoids algorithm

Genetic algorithms [7] have been applied to k-medoids algorithm [15]. In [15], the genetic k-medoids algorithm, GCA (genetic clustering algorithm) generates P individuals initially and each individual consists of k different parameters selected from T objects randomly. The fitness function is the inverse of the total distortion. The fitness is also modified using a linear scaling technique. The modified fitness of each individual is evaluated and a pair of individuals is selected based on the roulette selection. These two individuals are used in a crossover operation to generate temporary individuals with half the parameters from each selected individual. A mutation technique is then applied to this temporary individual. After obtaining the same population size, the evaluation, selection, crossover and mutation are applied again until the maximum number of generations is reached or the satisfied fitness is obtained. The experiments have been carried out to test the performance of GCA and $CLARA$ algorithms. Experimental results demonstrate that GCA is superior to $CLARA$ for a large number of medoids. For small number of medoids, both $CLARA$ and GCA find acceptable solutions.

2.7 *MCMRS* Sampling Scheme (Multi-Centroid, Multi-Run Sampling Scheme)

For k-means, each cluster is represented by the mean value of the objects in the cluster whereas each cluster is represented by one of the objects located near the centre of the cluster in the k-medoids algorithm. k-means can be sensitive while k-medoids is generally more robust to outliers (or noise). One of the main factors to limit the use of the k-medoids algorithm is the inefficiency of k-medoids algorithms comparing with k-means - k-means algorithm can be several orders of magnitude faster than the k-medoids algorithm. This drawback can be overcome with the aid of an efficient sampling scheme. From our empirical observations, we noticed that there is a higher probability of better medoids being selected within some distance from the centroid of the clusters. Based on this observation and the efficiency of the centroid-based clustering, we can generate k clusters of medoid candidates with each cluster containing $NumCandidate$ nearest objects from the centroid for each centroid-based cluster. k medoids can be collected from each object in each cluster randomly. This process iterates $NumSample$ times. This sampling scheme can be made more robust by repeating the above procedure many times. The proposed $MCMRS$ can be depicted as follows:

1. Repeat the following steps for $NumRun$ times.
2. Obtain representative centroids by calling the centroid-based clustering algorithm (such as k-means or GLA [14]) with random initialisation.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

3. Obtain $NumCandidate$ objects for each cluster by sorting the $NumCandidate$ nearest objects from the centroid in each cluster.
4. Repeat the following steps for $NumSample$ times.
5. Generate medoids by selecting one object from the $NumCandidate$ nearest objects in each cluster.
6. Calculate the average distance per object and update the best medoids.

3 *IMCMRS* (Incremental Multi-Centroid, Multi-Run Sampling Scheme)

Although $MCMRS$ [2] improves the efficiency and effectiveness of the k-medoids algorithms, it can be further improved by adopting the adaptive concept. The idea of Incremental Multi-Centroid, Multi-Run Sampling scheme ($IMCMRS$) is based on the observation that better medoids are not always near the centres of the clusters for each run of the centroid-based clustering algorithm. In order to reduce computation time, the sampling times and the number of candidate objects should be increased for the better results of centroid-based clustering and conversely reduced for the worse than average results. Based on this observation and the efficiency of the centroid-based clustering, we can generate $NumRun$ groups of medoid candidates with each group containing several nearest objects from the centroid for each centroid-based cluster. k medoids can be collected from each object in each group randomly. This process iterates $NumSample$ times. The groups with the worse results are deleted and more objects are chosen from the better groups. The $IMCMRS$ sampling scheme can be described as follows:

1. Obtain $NumRun$ groups representative centroids by calling the centroid-based clustering algorithm (such as k-means or GLA [14]) with random initialisation for $NumRun$ times.
2. Obtain the nearest object to the centroid for each cluster and choose the nearest objects as the medoids. Calculate the average distance per object. Set $n = 1$, where n is the iteration count.
3. Set $NumRun = NumRun/2$ and choose $NumRun$ groups with better average distance and obtain $O = NumCandidate+NumSelectxn$ objects for each cluster by sorting the O nearest objects from the centroid in each cluster.
4. Repeat the following step for $NumSample + StepSamplexn$ times.
5. Generate medoids by selecting one object from the nearest objects in each cluster.
6. Calculate the average distance per object and update the best medoids. If $NumRun = 1$, terminate the program; otherwise increment n and go to step 3.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

4 Experimental Results

4.1 Databases

Four artificial databases and one real image dataset were used for the experiments as follows:

1. 1,500 objects collected from four elliptic clusters.
2. 12,000 objects collected from twelve elliptic clusters.
3. 3,100 objects collected from five compact clusters.
4. 3,000 objects with 8 dimensions are generated from the Gauss-Markov source that is of the form $y_n = \alpha y_{n-1} + w_n$ where w_n is a zero-mean, unit variance, Gaussian white noise process, with $\alpha = 0.5$.
5. 16,384 objects with 16 dimensions are generated from the LENA grey-level image with size 512 by 512.

Space precludes the reporting of all results here (in particular, the LENA tests are not reported at all) but the associated report [3] and other related work (available at `http://kdm.first.flinders.edu.au`) contains full results.

4.2 Experimental results

Experiments were carried out to test the number of distances calculation and the average distance per object for the $CLARA$, $CLARANS$, $MCMRS$ and the proposed $IMCMRS$ algorithms. Since the computation time depends not only on the clustering algorithm but also on the use of computation facility. It is better to choose one measure criterion so that the measure results are the same for all types of computers and this measure criterion is proportional to the computation time. That is why we choose the number of distance calculation as the benchmark. Squared Euclidean distance measure is used in this paper. The four elliptic clusters were used for the first experiment and 12 medoids are selected from 1500 objects. For $CLARA$, the parameter q was set to 5 and s was set to $160 + 2k$. For $CLARANS$, the parameter *numlocal* was set to 5 and parameter *maxneighbor* was set to 270 (ie. 1.5% of $K \times t$). For $MCMRS$, k-means is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means is also used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and $StepSample$ in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. The experimental results based on 10 runs for $CLARA$, $CLARANS$, $MCMRS$ and $IMCMRS$ are shown in Table 1 and Fig. 1. In comparison with $CLARA$ and $CLARANS$, $IMCMRS$ may reduce the computation time by more than 91% and 80%. Both $MCMRS$ and $IMCMRS$ performed better than $CLARANS$ and $CLARA$ both in the computation time and the average distance per object.

The twelve elliptic clusters were used for the second experiment. 12 medoids are selected from 12000 objects. For $CLARA$, the parameter q was set to 5

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

seed	*CLARA*		*CLARANS*		*MCMRS*		*IMCMRS*	
	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)
1	0.228	2004	0.228	1609	0.216	808	0.213	197
2	0.221	2302	0.232	1067	0.217	821	0.212	221
3	0.236	2388	0.236	1351	0.217	825	0.212	197
4	0.230	2686	0.233	1154	0.218	811	0.213	249
5	0.227	2430	0.232	1572	0.218	907	0.213	199
6	0.237	2260	0.227	1223	0.216	803	0.212	223
7	0.232	2646	0.234	895	0.217	821	0.213	244
8	0.236	2516	0.231	1429	0.217	803	0.213	232
9	0.233	2345	0.231	1148	0.216	803	0.213	194
10	0.238	2728	0.231	1167	0.218	805	0.212	215
Ave.	0.232	2431	0.232	1140	0.217	811	0.213	217

Table 1. Results of Experiment for four elliptic clusters

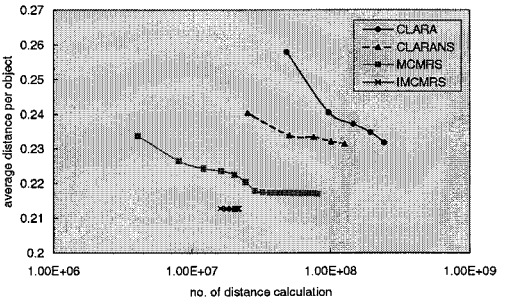

Fig. 1. Performance comparison for four elliptic clusters

and s was set to $960 + 2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* are set to 5 and 1800, respectively. For $MCMRS$, k-means is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means is also used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and $StepSample$ in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. As shown in Table 2, $IMCMRS$ will reduce the computation time by more than 98%, 97% and 73% by in comparison with $CLARA$, $CLARANS$ and $MCMRS$. Both $MCMRS$ and $IMCMRS$ are more efficient and effective than $CLARA$ and $CLARANS$.

The compact clusters with noise were used for the third experiment. 5 medoids are selected from 3100 objects. For $CLARA$, the parameter q was set to 5 and s was set to $200 + 2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* are set to 5 and 200, respectively. For $MCMRS$, k-means algorithm is used to generate 5 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means is also used to generate 5 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and $StepSample$ in $IMCMRS$ were set to 32, 1,

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

seed	*CLARA*		*CLARANS*		*MCMRS*		*IMCMRS*	
	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)
1	0.940	115659	0.931	91792	0.922	6587	0.927	1707
2	0.942	92528	0.943	63512	0.920	6554	0.920	1848
3	0.956	122462	0.935	76873	0.930	6496	0.920	1746
4	0.951	103413	0.928	79582	0.928	6447	0.939	1729
5	0.946	111577	0.949	90014	0.921	6483	0.918	1772
6	0.960	108856	0.936	72069	0.922	6632	0.919	1726
7	0.972	77562	0.936	84914	0.920	6426	0.918	1746
8	0.944	93889	0.934	93092	0.920	6549	0.920	1807
9	0.954	84365	0.930	59242	0.923	6573	0.922	1726
10	0.942	84365	0.931	75053	0.923	6541	0.920	1658
Ave.	0.951	99467	0.935	78615	0.923	6529	0.922	1747

Table 2. Results of Experiment for twelve elliptic clusters

1, 1 and 5 respectively. Experimental results based on 10 runs for *CLARA*, *CLARANS*, *MCMRS* and *IMCMRS* are shown in Table 3. If the database is not large and the medoid size is small, the performance of *CLARA* is better than *CLARANS* shown in Fig. 2. Comparison with *CLARA*, *CLARANS* and *MCMRS* show that *IMCMRS* may reduce the computation time by more than 60%, 81% and 83%, respectively.

seed	*CLARA*		*CLARANS*		*MCMRS*		*IMCMRS*	
	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)
1	2.436	253	2.432	584	2.398	646	2.397	99
2	2.425	274	2.457	600	2.397	647	2.397	107
3	2.430	264	2.429	604	2.398	655	2.397	96
4	2.440	295	2.442	504	2.397	649	2.397	102
5	2.405	232	2.431	583	2.398	651	2.397	103
6	2.411	253	2.419	606	2.398	655	2.397	102
7	2.435	243	2.457	530	2.397	642	2.397	104
8	2.422	285	2.470	519	2.397	654	2.397	104
9	2.444	253	2.417	620	2.397	647	2.397	112
10	2.440	274	2.424	581	2.398	645	2.397	120
Ave.	2.429	263	2.438	573	2.398	649	2.397	105

Table 3. Results of Experiment for compact clusters

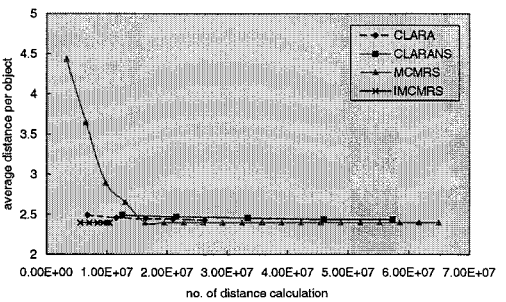

Fig. 2. Performance comparison for five compact clusters

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

A Gauss-Markov source was used for the fourth experiment. 32 medoids are selected from 3000 objects. For $CLARA$, the parameter q was set to 5 and s was set to $320 + 2k$. For $CLARANS$, the parameters *numlocal* and *maxneighbor* were set to 5 and 1200, respectively. For $MCMRS$, k-means was used to generate 32 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $IMCMRS$, k-means was again used to generate 32 centroid-based clusters. The parameters $NumRun$, $NumCandidate$, $NumSample$, $NumSelect$ and $StepSample$ in $IMCMRS$ were set to 32, 1, 1, 1 and 5 respectively. Experimental results shown in Table 4, compared with $CLARA$ and $MCMRS$, shows that $IMCMRS$ can reduce the computational complexity by more than 99% and 70%, respectively. The proposed $IMCMRS$ can reduce the computation time by approximately a factor of 30 and also obtains better average distance in comparison with $CLARANS$.

seed	$CLARA$		$CLARANS$		$MCMRS$		$IMCMRS$	
	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)	Ave. distance	Count of dist. (10^5)
1	4.559	154809	4.432	37604	4.487	4439	4.357	1340
2	4.592	163692	4.359	53234	4.476	4349	4.382	1330
3	4.551	178918	4.381	37512	4.490	4411	4.376	1273
4	4.578	172574	4.398	40559	4.495	4417	4.371	1364
5	4.559	182725	4.367	39694	4.489	4403	4.375	1234
6	4.526	167498	4.384	41370	4.489	4357	4.379	1278
7	4.527	185263	4.380	32312	4.485	4379	4.377	1309
8	4.483	162423	4.394	39600	4.499	4382	4.361	1307
9	4.545	190338	4.377	36707	4.491	4400	4.369	1276
10	4.514	180187	4.406	35835	4.485	4378	4.329	1309
Ave.	4.543	173843	4.388	39443	4.489	4391	4.368	1302

Table 4. Results of Experiment for Gauss-Markov source

5 Conclusions

In this paper, an incremental sampling scheme using multiple centroids with multiple runs ($IMCMRS$) is presented. This sampling scheme can be applied to PAM, $CLARA$, $CLARANS$ and $CLASA$. Experimental results based on four artificial databases and one real image dataset confirms that the proposed $IMCMRS$ not only can reduce the average distance but also speed the clustering process. The computation load in $IMCMRS$ can be further improved by applying a more efficient centroid-based clustering method [9].

References

[1] S. C. Chu, J. F. Roddick, and J. S. Pan. A comparative study and extensions to k-medoids algorithms. In *Fifth International Conference on Optimization : Techniques and Applications*, pages 1708–1717, Hong Kong, China, 2001.

[2] S-C. Chu, J. F. Roddick, and J. S. Pan. Efficient k-medoids algorithms using multi-centroids with multi-runs sampling scheme. In *Workshop on Mining Data for CRM*, Taipei, Taiwan, 2002. Springer.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

[3] S-C. Chu, J. F. Roddick, and J. S. Pan. An incremental multi-centroid, multi-run sampling scheme for k-medoids-based algorithms - extended report. Technical Report KDM-02-003, KDM Laboratory, Flinders University, 2002.

[4] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, OR, 1996. AAAI Press.

[5] V. Estivill-Castro and I. Lee. Autoclust+: Automatic clustering of point-data sets in the presence of obstacles. In J.F. Roddick and K. Hornsby, editors, *International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining, TSDM2000*, volume 2007 of *Lecture Notes in Artificial Intelligence*, Lyon, France, 2000. Springer.

[6] V. Estivill-Castro and I. Lee. Autoclust: Automatic clustering via boundary extraction for massive point-data sets. In *Fifth International Conference on Geocomputation*, 2000.

[7] D. E. Goldberg. *Genetic algorithm in search, optimization and machine learning.* Addison-Wesley, 1989.

[8] S. Guha, Rajeev Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD International Conference on the Management of Data*, pages 73–84, Seattle, WA, USA, 1998.

[9] H. C. Huang, J. S. Pan, Z. M. Lu, S. H. Sun, and H. M. Hang. Vector quantization based on genetic simulated annealing. *Signal Processing*, 81(7):1513–1523, 2001.

[10] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: a hierarchical clustering algorithm using dynamic modeling. *Computer*, 32:32–68, 1999.

[11] L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis.* John Wiley and Sons, New York, 1990.

[12] S. Kirkpatrick, C.D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[13] R. Krishnapuram, A. Joshi, and L. Yi. A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering. In *IEEE International Fuzzy Systems Conference*, pages 1281–1286, Seoul, Korea, 1999.

[14] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[15] C. B. Lucasius, A. D. Dane, and G. Kateman. On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison. *Analytica Chimica Acta*, pages 647–669, 1993.

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley symposium on mathematics, statistics and Probability*, volume 1, pages 281–296, 1967.

[17] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, *Twentieth International Conference on Very Large Data Bases*, pages 144–155, Santiago, Chile, 1994. Morgan Kaufmann.

[18] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient clustering method for very large databases. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 103–114, Montreal, 1996.